Docket No.: 50023-135

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	
Hideaki MIYAKE, et al.	:	
Serial No.: 09/812,894	:	Group Art Unit: 2133
Filed: March 21, 2001	:	Examiner:

For: DEBUGGING SUPPORTING APPARATUS, DEBUGGING SUPPORTING METHOD AND RECORDING MEDIUM READABLE BY COMPUTER WITH ITS PROGRAMS RECORDED THEREON

TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENTS

Honorable Commissioner for Patents and Trademarks
Washington, D. C. 20231

Sir:

At the time the above application was filed, priority was claimed based on the following applications:

Japanese Patent Application No. 2000-094064, Filed March 30, 2000; and

Japanese Patent Application No. 2000-375641, December 11, 2000

A copy of each priority application listed above is enclosed.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

Stephen A. Becker
Registration No. 26,527

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 SAB:ykg
Date: June 4, 2001
Facsimile: (202) 756-8087

日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２０００年　３月３０日

出願番号
Application Number: 特願２０００－０９４０６４

出願人
Applicant (s): 松下電器産業株式会社

２０００年１２月１５日

特許庁長官
Commissioner,
Patent Office 及川耕造

出証番号　出証特２０００－３１０４１５４

【書類名】　特許願

【整理番号】　2022510576

【提出日】　平成12年 3月30日

【あて先】　特許庁長官　殿

【国際特許分類】　G06F 11/28

【発明者】

【住所又は居所】　東広島市鏡山３丁目１０番１８号　株式会社松下電器情報システム広島研究所内

【氏名】　三宅　秀明

【発明者】

【住所又は居所】　東広島市鏡山３丁目１０番１８号　株式会社松下電器情報システム広島研究所内

【氏名】　枯木　正吉

【発明者】

【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内

【氏名】　鶴本　克己

【特許出願人】

【識別番号】　000005821

【氏名又は名称】　松下電器産業株式会社

【代理人】

【識別番号】　100083172

【弁理士】

【氏名又は名称】　福井　豊明

【手数料の表示】

【予納台帳番号】　009483

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9713946

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　デバッグ支援システムおよびデバッグ支援方法

【特許請求の範囲】

【請求項１】　ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際、ＯＳシミュレータが管理するＯＳ管理情報をＯＳデバッガから操作できるようにしたデバッグ支援システムにおいて、

上記ＯＳシミュレータの要請に応じてＯＳ管理情報が書き込まれるとともに、上記ＯＳデバッガの要請に応じてＯＳ管理情報が読み出される共有ファイルを備えたことを特徴とするデバッグ支援システム。

【請求項２】　上記共有ファイルにＯＳ管理情報を書き込むＯＳ管理情報書き込み手段をアプリケーション側に備えるとともに、

上記共有ファイルからＯＳ管理情報を読み出すＯＳ管理情報読み出し手段をＯＳデバッガ側に備えた請求項１に記載のデバッグ支援システム。

【請求項３】　上記ＯＳ管理情報読み出し手段が、任意のタイミングで共有ファイルからＯＳ管理情報を読み出す請求項２に記載のデバッグ支援システム。

【請求項４】　上記ＯＳ管理情報読み出し手段が、所定の周期で共有ファイルからＯＳ管理情報を読み出す請求項２に記載のデバッグ支援システム。

【請求項５】　ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際、ＯＳシミュレータが管理するＯＳ管理情報をＯＳデバッガから操作できるようにしたデバッグ支援システムにおいて、

上記ＯＳデバッガの要請に応じてＯＳ管理情報を変更するためのデバッグ命令が書き込まれるとともに、上記アプリケーションの要請に応じてデバッグ命令が読み出される共有ファイルを備えたことを特徴とするデバッグ支援システム。

【請求項６】　上記共有ファイルにデバッグ命令を書き込むデバッグ命令書き込み手段をＯＳデバッガ側に備えるとともに、

上記共有ファイルからデバッグ命令を読み出すデバッグ命令読み出し手段をアプリケーション側に備えた請求項５に記載のデバッグ支援システム。

【請求項７】　ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際に用いるデバッグ支援システムにおいて、

上記ＯＳシミュレータが管理するＯＳ管理情報を変更するためのデバッグ命令が書き込まれた共有ファイルと、該共有ファイルからデバッグ命令を読み出すデバッグ命令読み出し手段とを備えたことを特徴とするデバッグ支援システム。

【請求項８】　上記デバッグ命令読み出し手段が、任意のタイミングで共有ファイルからデバッグ命令を読み出す請求項６又は７に記載のデバッグ支援システム。

【請求項９】　上記デバッグ命令読み出し手段が、所定の周期で共有ファイルからデバッグ命令を読み出す請求項６又は７に記載のデバッグ支援システム。

【請求項１０】　ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際、ＯＳシミュレータが管理するＯＳ管理情報をＯＳデバッガから操作できるようにしたデバッグ支援方法において、

ＯＳシミュレータとＯＳデバッガとの間に共有ファイルを介在させることによって上記操作を可能としたことを特徴とするデバッグ支援方法。

【請求項１１】　ＯＳシミュレータが管理するＯＳ管理情報をＯＳデバッガから操作しながら、ＯＳシミュレータによるシミュレーション環境で動作するアプリケーションを作成する方法において、

ＯＳシミュレータとＯＳデバッガとの間に共有ファイルを介在させることによって上記操作を可能としたことを特徴とするアプリケーションの作成方法。

【請求項１２】　ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際、

ＯＳシミュレータとＯＳデバッガとの間に共有ファイルを介在させることによって、ＯＳシミュレータが管理するＯＳ管理情報をＯＳデバッガから操作できるようにためのプログラムをコンピュータ読み取り可能な形態で記録したことを特徴とする記録媒体。

【発明の詳細な説明】

【発明の属する技術分野】

本発明は、デバッグ支援システムに関し、特に、ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションをデバッグする際に用いるデバッグ支援システムに関するものである。

【従来の技術】

近年の携帯電話など（以下「機器」という）には、図４（ａ）に示すように、様々な機能を実現するためのアプリケーション１０が組み込まれており、このアプリケーション１０は、通常、機器専用のオペレーティングシステム（以下「専用ＯＳ」という）２０ａ上で動作するようになっている。従って、このような組み込み型のアプリケーション１０の開発は、機器のハードウェアや専用ＯＳ２０ａの開発と並行して行えるよう、図４（ｂ）に示すように、汎用コンピュータ上のマルチプロセス環境（すなわち汎用ＯＳ５０上）に構築した組み込み先のシミュレーション環境において行う。

上記シミュレーション環境は、専用ＯＳ２０ａの動作（例えば、タスク管理動作や、イベントフラグなどのリソース管理動作）をシミュレートしたＯＳシミュレータ２０をライブラリという形式でアプリケーション１０にリンクさせることによって構築することができる。

ここで、アプリケーション１０のデバッグは、汎用のアプリケーション開発ツール（図示せず）に含まれるアプリケーションデバッガ３０を用いて行うようになっている。このデバッグとは、プログラム中のバグを取り除く作業をいい、デバッガとは、デバッグを効率よく行えるようにしたユーティリティプログラムをいう。

しかしながら、上記シミュレーション環境で開発したアプリケーション１０をデバッグする場合、アプリケーション１０の実行中におけるタスクの状態や、このタスクの状態を変化させるイベントの有無を１または０のビットパターンで表したイベントフラグなど、図示しないメモリにおいてＯＳシミュレータ２０が管理する各種情報（以下「ＯＳ管理情報」という）を参照あるいは変更したいときがある。すなわち、アプリケーションデバッガ３０は、上記したように汎用のアプリケーション開発ツールであるため、ＯＳ管理情報を参照あるいは変更（以下、両者を一括して「操作」という）するという特別な機能は備えていない。

そこで、最近では、アプリケーションデバッガ３０だけでなく、ＯＳ管理情報を操作するためのユーティリティプログラム（以下「ＯＳデバッガ４０」という）を備えたデバッグ支援システムが登場している。

このＯＳデバッガ４０は、図３に示すように、ＯＳシミュレータ２０とは別のプロセス空間に存在するため、ＯＳ管理情報に直接アクセスすることはできない。従って、アプリケーション１０側とＯＳデバッガ４０側の両方に通信手段C_1・C_2を備え、これら通信手段C_1・C_2がプロセス間通信を行うことによって、ＯＳデバッガ４０からＯＳ管理情報を操作できるようにしている。

【発明が解決しようとする課題】

ところで、ブレークポイントを設定するなどしてアプリケーション１０を停止させ、この時点のＯＳ管理情報を参照できれば、バグの位置を見当付けるのが容易になる。

しかしながら、アプリケーション１０が停止した場合は、このアプリケーション１０とリンクしているＯＳシミュレータ２０も停止する。すなわち、上記従来のデバッグ支援システムのようにプロセス間通信を採用した構成によれば、アプリケーション１０が停止すると、ＯＳシミュレータ２０とＯＳデバッガ４０との間で通信が行えなくなり、この時点のＯＳ管理情報を参照できないという問題があった。

この問題は、ブレークポイントを設定してアプリケーション１０を意識的に停止させた場合だけでなく、アプリケーション１０がハングアップした場合も同様に発生する（ハングアップしたの時点のＯＳ管理情報を参照できなければ、その原因を特定するのは難しい）。

本発明は上記従来の事情に基づいて提案されたものであって、ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションが停止した場合でも、この時点のＯＳ管理情報を参照できるようにしたデバッグ支援システムを提供することを目的とするものである。

【課題を解決するための手段】

本発明は上記目的を達成するために以下の手段を採用している。すなわち、本発明は、図１に示すように、ＯＳシミュレータ２０によるシミュレーション環境で開発したアプリケーション１０をデバッグする際、ＯＳシミュレータ２０が管理するＯＳ管理情報をＯＳデバッガ４０から操作できるようにしたデバッグ支援システムを前提としている。

ここで、共有ファイルＦには、上記ＯＳシミュレータ２０の要請に応じてＯＳ管理情報が書き込まれ、このＯＳ管理情報は、上記ＯＳデバッガ４０の要請に応じて読み出される。このようにすれば、アプリケーション１０が停止した場合でも、この時点のＯＳ管理情報を上記共有ファイルＦから参照することができる。

また、図２に示すように、ＯＳ管理情報を変更するためのデバッグ命令を共有ファイルＦに書き込むとともに、この共有ファイルＦからデバッグ命令を読み出すようにすれば、ＯＳ管理情報を変更できる。

【発明の実施の形態】

以下に本発明の実施の形態を図面に従って詳細に説明する。

（第１の実施の形態）

図１は、本発明を適用したデバッグ支援システムの概略機能ブロック図であり、以下その構成を上記従来と異なる点のみ説明する。

まず、図示しないアプリケーションデバッガ３０を用いてアプリケーション１０を実行すると、例えば、タスクＡがシステムコール「set __flg 」を発行し、このシステムコール「set __flg 」を受け付けたＯＳシミュレータ２０のシステムコール処理手段２１は、イベントフラグの値を０から１に変更するようＯＳ管理情報変更手段２２に指示する。

これによって、ＯＳ管理情報変更手段２２は、図示しないメモリに格納されているイベントフラグの値を０から１に変更した後、この変更内容をアプリケーション１０に含まれるＯＳ管理情報書き込み手段１２に通知し、この通知を受けたＯＳ管理情報書き込み手段１２は、イベントフラグの値が０から１に変更された旨を共有ファイルＦに書き込む。なお、共有ファイルＦとは、複数のプロセスによって同時に共有されるファイルをいう。

ここで、アプリケーション１０が停止し、この時点のＯＳ管理情報を参照したい場合は、ＯＳデバッガ４０のユーザインターフェイス４１を介して「イベントフラグの値参照」を指示する。

これによって、メイン処理手段４２は、イベントフラグの値を読み出すようＯＳ管理情報読み出し手段４３に指示し、この指示を受けたＯＳ管理情報読み出し手段４３は、共有ファイルＦからイベントフラグの値を読み出して表示処理手段

４４に渡し、更に、表示処理手段４４は、このイベントフラグの値をユーザが視覚的に理解できるような形式に変換して表示する。

以上のように、本発明を適用したデバッグ支援システムによれば、イベントフラグの値を共有ファイルＦに書き込むようにしているため、アプリケーション１０が停止した場合でも、この時点のイベントフラグの値を参照することができる。

なお、イベントフラグの値を共有ファイルＦから読み出すタイミングは、「イベントフラグの値参照」が指示されたときに限定されるものではない。すなわち、ＯＳ管理情報読み出し手段４３に対して周期的に読み出しを指示する手段を備えた構成としてもよい。このようにすれば、ＯＳ管理情報の時間的遷移を知ることができるため、予期しないアプリケーション１０の状態を容易に発見することができる。

また、ＯＳ管理情報の種別毎に共有ファイルＦの使用領域を決めておけば、イベントフラグの値だけでなく、他のＯＳ管理情報（タスクの状態・セマフォの状態・メールボックスの状態など）も同様の手順で参照できることはいうまでもない。

（第２の実施の形態）

ところで、上記のように参照したイベントフラグの値が１である場合は、バグがないものとしてアプリケーション１０の実行を再開すればよいが、０である場合は何らかのバグがあり、このバグを取り除く必要がある。

しかしながら、上記デバッグが完了するまで残りの動作検証が行えないのでは効率が悪い。すなわち、まずはアプリケーション１０の全部について動作検証をし、その後、一括してデバッグするのが好ましい。

そこで、本実施の形態では、ＯＳ管理情報が不正な値となった場合は、このＯＳ管理情報をＯＳデバッガ４０から変更できるようにしており、以下その構成を図２に従って説明する。

例えば、イベントフラグの値を０から１に変更したい場合は、ＯＳデバッガ４０のユーザインターフェイス４１を介して「デバッグ命令」を指示し、更に、このデバッグ命令のうちの「イベントフラグの値変更（変更後の値：１）」を指示

する。これによって、メイン処理手段４２は、イベントフラグの値を０から１に変更するようデバッグ命令書き込み手段４５に指示し、この指示を受けたデバッグ命令書き込み手段４５は、イベントフラグの値を０から１に変更するシステムコール「set __flg 」を、それに対応するデバッグ命令機能コード(-48) に変換して共有ファイルＦに書き込む。

一方、アプリケーション１０に含まれるデバッグ命令読み出し手段１５は、タイマ１６によって共有ファイルＦを周期的に参照するようになっており、上記のようにデバッグ命令機能コード(-48) が共有ファイルＦに書き込まれると、このデバッグ命令機能コード(-48) を読み出してシステムコール情報生成手段１４に渡す。

ここで、システムコール情報生成手段１４は、上記デバッグ命令機能コード(-48) に対応するシステムコール情報「set __flg 」を生成してデバッグタスク起動手段１３に渡し、更に、デバッグタスク起動手段１３は、このシステムコール情報「set __flg 」を指定してデバッグタスク１７を起動する。なお、システムコール情報「set __flg 」を指定する方法は、デバッグタスク起動のパラメータや共有ファイルＦを用いる方法など様々あり、特に限定されるものではない。

その後、上記のように起動されたデバッグタスク１７がＯＳシミュレータ２０に対してシステムコール「set __flg 」を発行し、これによって、メモリに格納されているイベントフラグの値が０から１に変更されると、ＯＳシミュレータ２０のタスク管理手段（図示せず）が、イベントフラグ待ちの状態にあるタスクＢを実行状態に遷移させる。

以上のように、本実施の形態によれば、何らかのバグが存在することによってＯＳ管理情報が不正な値となった場合でも、このＯＳ管理情報をＯＳデバッガ４０から変更することができる。また、このような変更機能を利用すれば、アプリケーション１０のプログラムを修正することなく、割り込みが発生した状況をつくることも可能である。

なお、ここでは、ＯＳデバッガ４０からデバッグ命令を指示することとしているが、本発明はこれに限定されるものではない。すなわち、予めデバッグ命令を共有ファイルＦに格納しておき、このデバッグ命令をデバッグ命令読み出し手段

１５が周期的に（或いは、任意のタイミングで）読み出すようにしてもよい。このようにすれば、ＯＳデバッガ４０を用いることなくＯＳ管理情報を変更できる。

また、デバッグ命令を実行する手段はデバッグタスク１７に限定されるものではない。すなわち、ここでは、デバッグタスク１７がシステムコールを発行することによってデバッグ命令が実行される動作を例示しているが、システムコールには、タスクからのみ発行可能・割り込みハンドラからのみ発行可能・この両方から発行可能という種別があるため、この種別に応じた実行手段を選択するようにしている。

更に、上記の説明では、デバッグタスク１７の状態遷移について特に言及していないが、システムコールを発行し終えたデバッグタスク１７は、消滅しても、常駐しても（すなわち、次のデバッグ命令が共有ファイルＦに書き込まれるまで待ち状態となっても）、いずれであってもかまわない。

【発明の効果】

以上のように、本発明によれば、効果的なデバッグを行うことができる。

すなわち、第１の実施の形態によれば、ＯＳ管理情報を共有ファイルに書き込むようにしているため、アプリケーションが停止した場合でも、この時点のＯＳ管理情報をＯＳデバッガから参照することができる。従って、バグの位置を見当付けるためにブレークポイントを設定することや、ハングアップの原因を容易に特定することが可能である。

また、周期的にＯＳ管理情報を参照するようにしておけば、予期しないアプリケーションの状態を容易に発見することができる。

一方、第２の実施の形態によれば、ＯＳ管理情報をＯＳデバッガから変更できるため、ＯＳ管理情報が不正な値となった場合でも、アプリケーションの動作検証を継続することや、アプリケーションのプログラムを修正することなく割り込みが発生した状況をつくることが可能である。

また、予めデバッグ命令を共有ファイルに格納しておけば、ＯＳデバッガを用いることなくＯＳ管理情報を変更することができる。

【図面の簡単な説明】

【図１】

第１の実施の形態におけるデバッグ支援システムの概略機能ブロック図。

【図２】

第２の実施の形態におけるデバッグ支援システムの概略機能ブロック図。

【図３】

従来におけるデバッグ支援システムの概略機能ブロック図。

【図４】

組み込み型アプリケーション１０の開発環境の説明図。

【符号の説明】

１０　アプリケーション１０

１２　ＯＳ管理情報書き込み手段

１３　デバッグタスク起動手段

１４　システムコール情報生成手段

１５　デバッグ命令読み出し手段

１７　デバッグタスク

２０　ＯＳシミュレータ２０

２１　システムコール処理手段２１

２２　ＯＳ管理情報変更手段

３０　アプリケーション１０デバッガ

４０　ＯＳデバッガ４０

４１　ユーザインターフェイス

４２　メイン処理手段

４３　ＯＳ管理情報読み出し手段

４４　表示処理手段

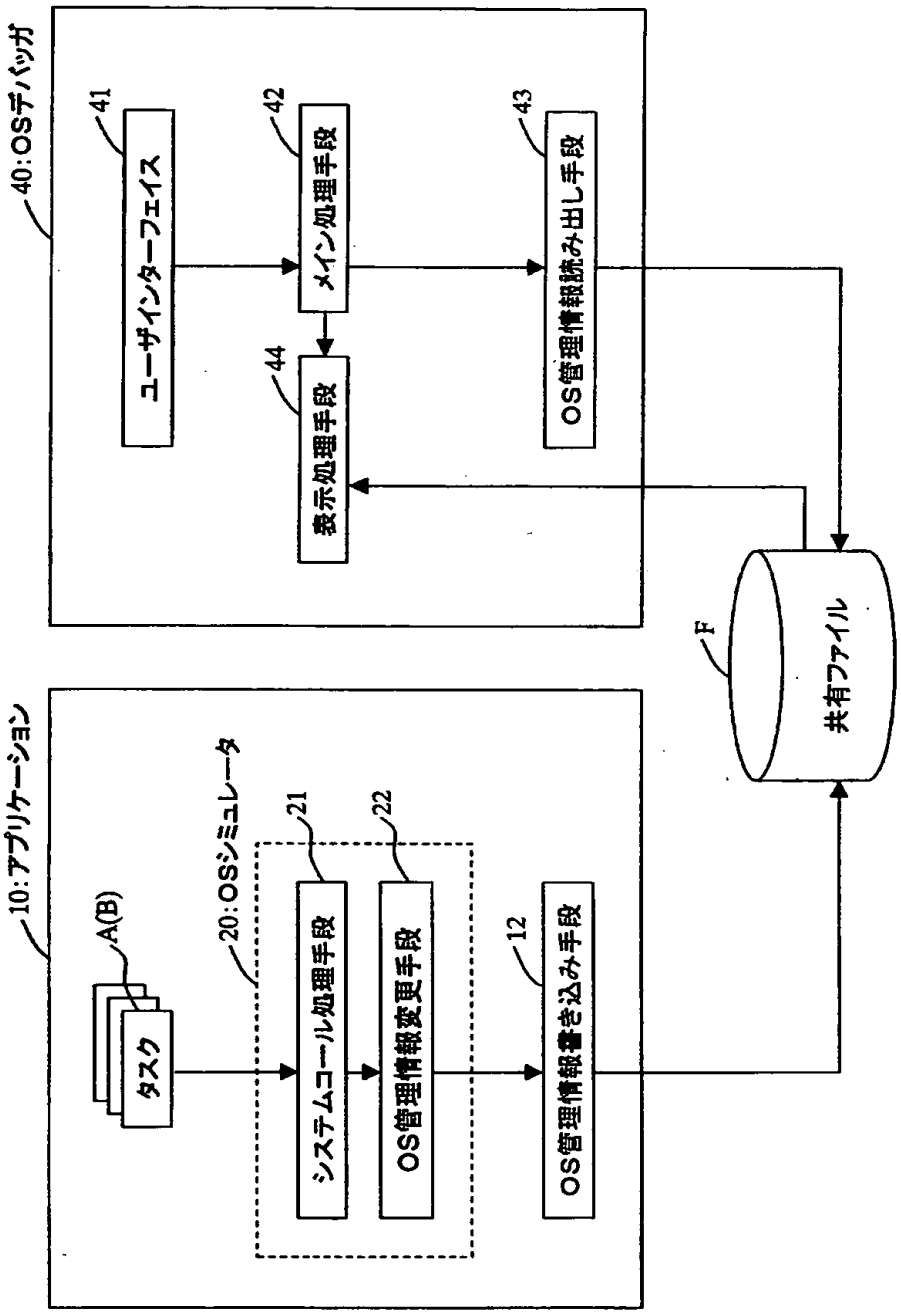
４５　デバッグ命令書き込み手段

Ａ，Ｂ　タスク

【書類名】　　図面

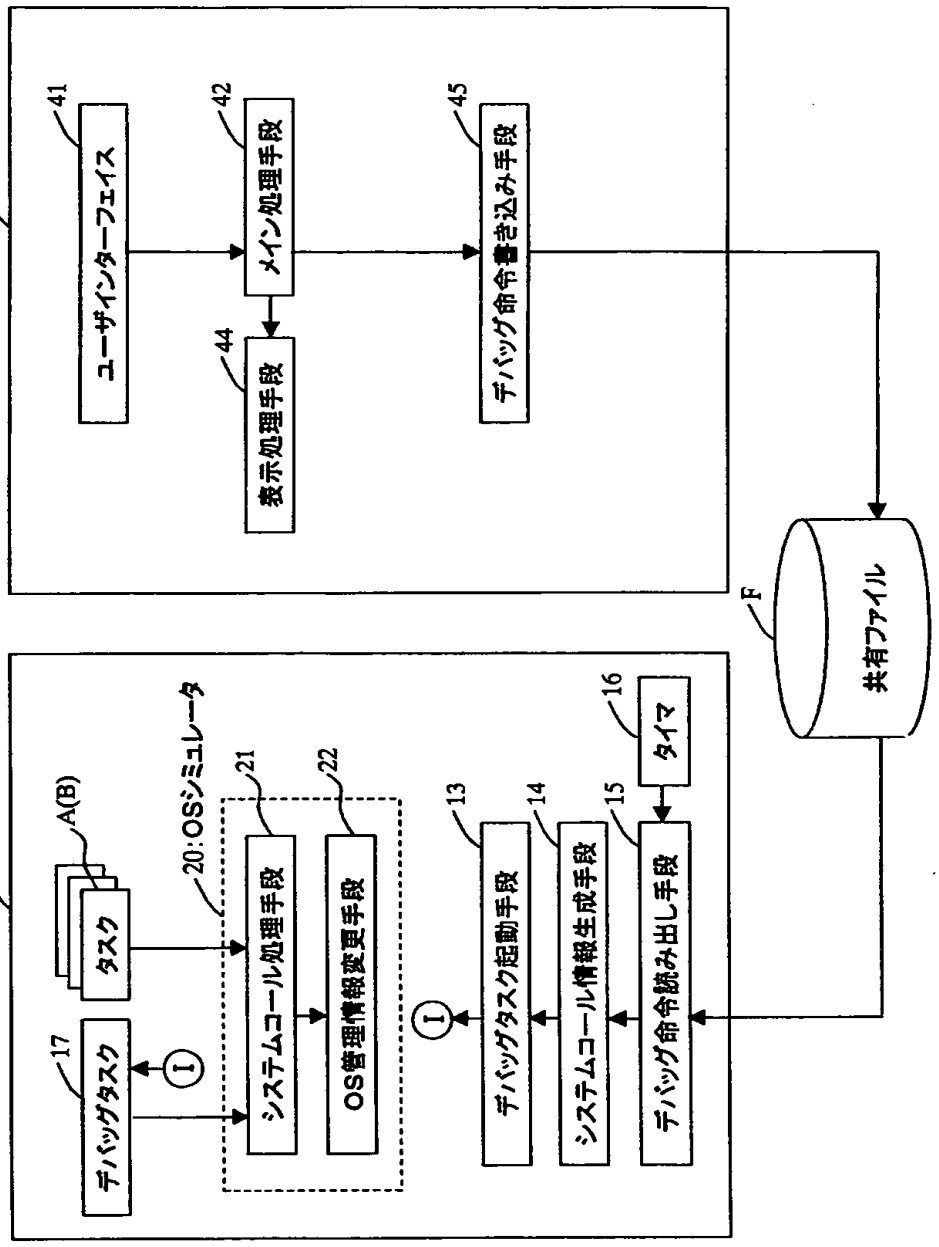
【図１】

第１の実施の形態におけるデバッグ支援システムの概略機能ブロック図



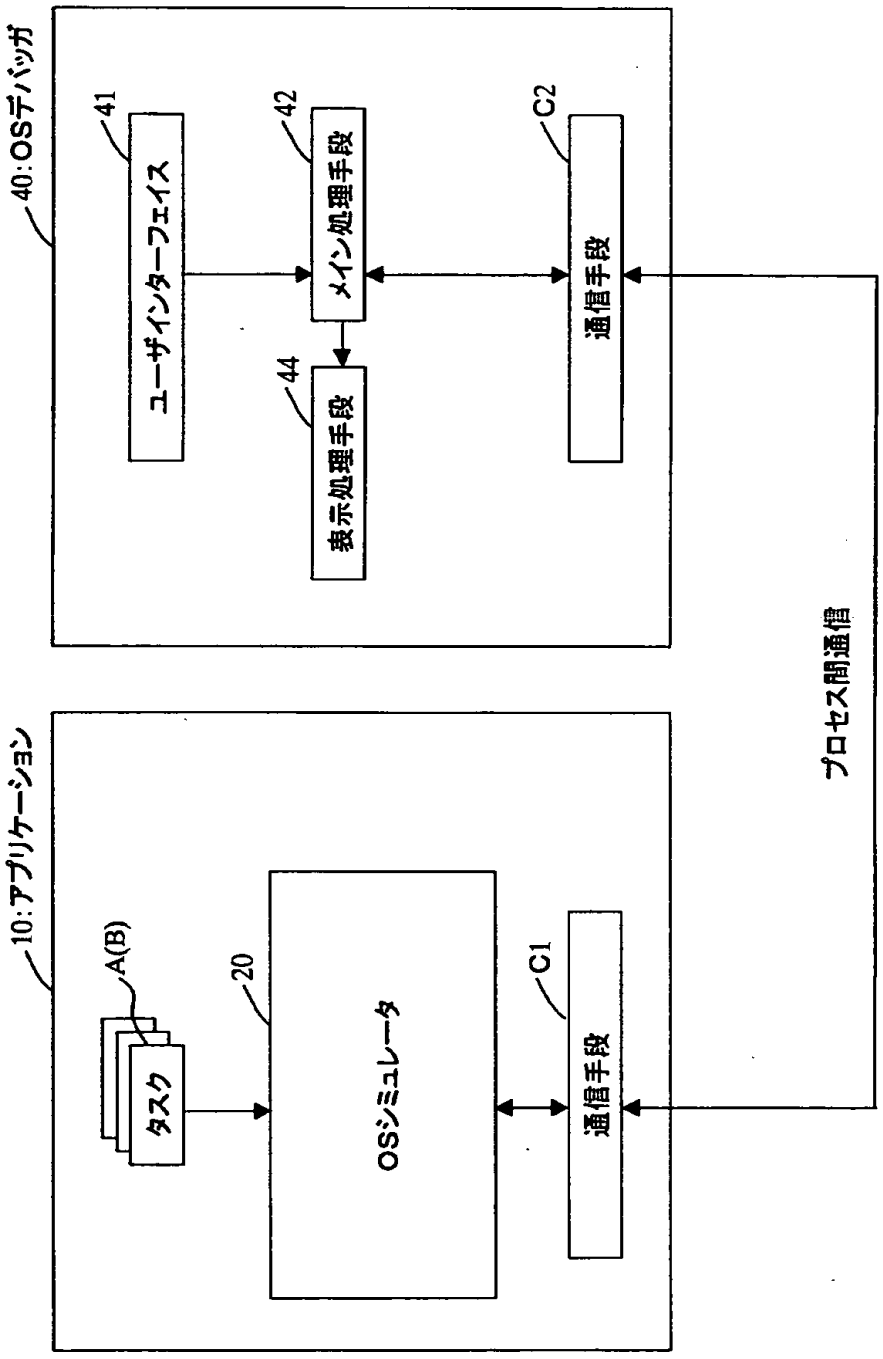
【図２】

第2の実施の形態におけるデバッグ支援システムの概略機能ブロック図



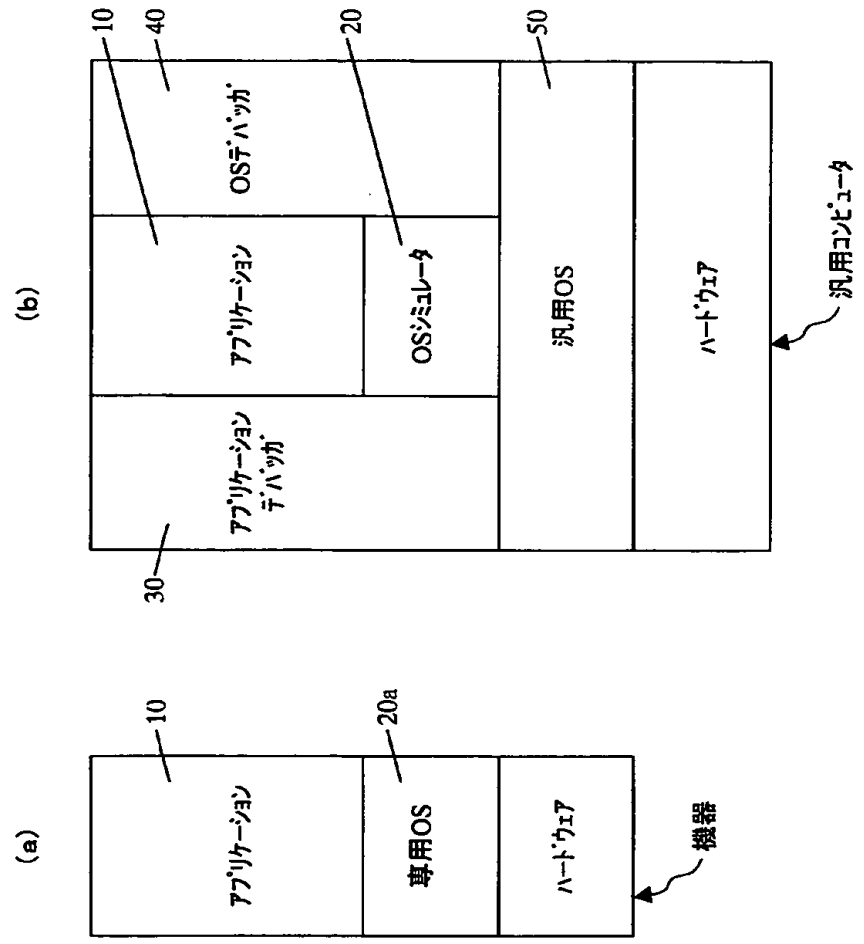
【図３】

従来におけるデバッグ支援システムの概略機能ブロック図



【図４】

組み込み型アプリケーションの開発環境



【書類名】　要約書

【要約】

【課題】　ＯＳシミュレータによるシミュレーション環境で開発したアプリケーションが停止した場合でも、この時点のＯＳ管理情報を参照できるようにしたデバッグ支援システムを提供する。

【解決手段】　共有ファイルＦには、ＯＳシミュレータ２０の要請に応じてＯＳ管理情報が書き込まれ、このＯＳ管理情報は、ＯＳデバッガ４０の要請に応じて読み出される。すなわち、アプリケーション１０側に備えたＯＳ管理情報書き込み手段１２が共有ファイルＦにＯＳ管理情報を書き込み、ＯＳデバッガ４０側に備えたＯＳ管理情報読み出し手段４３が共有ファイルＦからＯＳ管理情報を読み出すようになっている。このようにすれば、アプリケーション１０が停止した場合でも、この時点のＯＳ管理情報を共有ファイルＦから参照できる。

【選択図】　図１

出 願 人 履 歴 情 報

識別番号　〔０００００５８２１〕

１．変更年月日　１９９０年　８月２８日
［変更理由］　新規登録
住　所　大阪府門真市大字門真１００６番地
氏　名　松下電器産業株式会社